AMENDMENTS TO THE CLAIMS:

The following listing of claims supersedes all prior versions and listings of claims in this application:

1. (Currently Amended) A method of using a computer having a CPU linked to a digital data memory for processing requests for allocation of a memory block of a data memory, wherein segments of the data memory are associated with different levels according to [[their]] segment size, the method comprising:

(a) said CPU receiving a request for the allocation of a memory block within said digital data memory including receiving a binary data set indicative of requested memory block size wherein each bit of the binary data set is associated with an entry of a lookup table associated with one of said levels;

(b) said CPU determining the lowest of said levels containing a segment equal to or larger in size than the requested memory block including determining a most significant set bit of the binary data set, and determining from the lookup table entry associated with the most significant set bit a lowest of said levels containing a segment of a size equal to or larger than the requested memory block;

(c) said CPU determining, in the level determined in step (b), the availability of a free segment of a size equal to or larger than the requested memory block; and

(d) said CPU allocating a free segment depending on the determination in step (c).

2. (Previously Presented) The method of claim 1, further comprising:

(e) repeating steps (c) and (d) for a next higher level if no free segment of a size equal to or larger than the requested memory block has been found in step (c); and

(f) repeating step (e) until a free segment has been allocated or there is no next level.

3. (Previously Presented) The method of claim 1, wherein each level is associated with a different granule size to the power of two, and sizes of memory segments allocated to a level are related to the granule size of the respective level.

4. (Previously Presented) The method of claim 3, wherein the granule size associated with a level defines a size difference between memory segments allocated to that level.

5. (Previously Presented) The method of claim 4, wherein step (a) further comprises rounding the requested memory block to a lowest granule size before performing steps (b) to (d).

6. (Previously Presented) The method of claim 1, wherein each level is associated with a table of pointers indicative of memory addresses of free memory segments of a size allocated to the respective level.

7. (Previously Presented) The method of claim 1, wherein step (d) comprises returning a pointer to the allocated free segment.

8. (Previously Presented) The method of claim 1, wherein (d) comprises returning a null pointer if no free segment is allocated.

9. (Previously Presented) The method of claim 1, wherein a bitmap is indicative of a state of memory segments (free, allocated), the bitmap comprising a root bitmap, each bit of the root bitmap being indicative of whether or not an associated one of said levels contains at least one free segment, and wherein step (b) further comprises determining from the root bitmap said lowest level containing a segment of a size equal to or larger than the requested memory block.

10. (Cancelled)

11. (Previously Presented) The method of claim 9, wherein each mask of a set of predetermined masks is associated with one of said levels, and step (c) further comprises performing a logic operation on the mask associated with the lowest level determined in step (b) and said binary data set, wherein a result of said logic operation is an index to bits of the bitmap indicative of the state of a segment of a size equal to or larger than the requested memory block.

12. (Previously Presented) The method of claim 11, wherein said bitmap comprises a plurality of second and third stage bitmaps, each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps, each bit of said second stage bitmaps being indicative of a state of an associated predetermined number of bits of one of said third stage bitmaps, and each bit of the third stage bitmap being indicative of whether or not an associated segment is free, and wherein the operation result is an index to one bit of the second stage bitmap and one bit of said predetermined number of bits of the third stage bitmap associated with said one bit of the second stage bitmap, said one bit of the third stage bitmap being indicative of the state of a segment of a size equal to or larger than the requested memory block.

13. (Previously Presented) The method of claim 12, wherein step (c) further comprises, if no free segment is found, repeating the determination for a next more significant bit of said predetermined number of bits of the third stage bitmap, until a free segment is found or there is no more significant bit of said predetermined number of bits of the third stage bitmap.

14. (Original) The method of claim 13, wherein step (c) further comprises, if no free segment is found, repeating the determination for the predetermined number of bits of the third stage bitmap associated with the next more significant set bit of the second stage bitmap, until a free segment is found or there is no more significant bit of said one second stage bitmap.

15. (Original) The method of claim 14, wherein step (c) further comprises, if no free segment is found, repeating the determination for the second stage bitmap associated with the next more significant set bit of the root bitmap, until a free segment is found or there is no more significant bit of the root bitmap.

16. (Previously Presented) The method of claim 12, wherein each bit of the third stage bitmaps is associated with an entry in a table of pointers indicative of memory addresses of free memory segments.

17. (Previously Presented) A method of using a computer having a CPU linked to a digital data memory for managing a data memory, the method comprising:

said CPU defining a number of levels of the digital data memory;

said CPU defining a different range of memory segment sizes for each level; and

said CPU generating a lookup table, wherein each entry of the lookup table is associated with a bit of a binary data set that indicates size of a requested memory block, and wherein each entry of the lookup table indicates one of the levels, whereby a request for allocation of a memory block is processable by determining a level containing segments of a size equal to or larger than the requested memory block using the lookup table, and allocating a free segment of the same size or larger than the requested memory block in that level.

18. (Previously Presented) The method of claim 48, wherein the granule size defines a size difference between memory segments in each level.

19. (Previously Presented) The method of claim 17, further comprising:

generating a bitmap indicative of a state of each segment (free, allocated) and of whether or not a level contains at least one free segment.

20. (Original) The method of claim 19, wherein the bitmap comprises a root bitmap, each level being associated with one bit of the root-bitmap, and a plurality of second and third stage bitmaps associated with the segments, each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps, and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps.

21. (Previously Presented) The method of claim 19, further comprising:

updating the bitmap when a segment is allocated.

22. (Previously Presented) The method of claim 19, further comprising:

updating the bitmap when a segment is freed.

23. (Previously Presented) The method of claim 17, further comprising generating a table of pointers for each level indicative of memory addresses of free memory segments of a size associated with the respective level.

24. (Previously Presented) The method of claim 23, further comprising:

generating a bitmap indicative of a state of each segment (free, allocated) and of whether or not a level contains at least one free segment,

wherein each bit of the third stage bitmaps is associated with an entry in the tables of pointers.

25. (Cancelled)

26. (Previously Presented) The method of claim 17, further comprising:

generating a set of masks, wherein each of the set of masks is associated with one of said levels, and wherein a logical operation of a binary data set indicative of the size of the requested memory block and the mask associated with a level containing segments of the same size as or larger than the requested memory block results in an index to a segment of a size the same as or larger than the requested memory block in that level.

27. (Previously Presented) A method of claim 17, further comprising:

creating a first doubly linked list of consecutive memory segments irrespective of size and status (free, allocated); and

creating a second doubly linked list of free memory segments of the same size.

28. (Previously Presented) The method of claim 27, wherein memory segments in the first doubly linked list are arranged in order of associated memory addresses.

29. (Previously Presented) The method of claim 27, further comprising, when freeing a memory segment:

determining a state of memory segments adjacent to the memory segment to be freed using the first doubly linked list;

merging the memory segment to be freed with free adjacent memory segments; and

updating the first and second doubly linked lists accordingly.

30. (Previously Presented) The method of claim 27, wherein each second doubly linked list is a LIFO (Last In First Out) list.

31. (Previously Presented) The method of claim 27, comprising updating the second doubly linked list upon allocation of a memory segment upon request.

32. (Previously Presented) The method of claim 27, further comprising, if a segment determined for allocation upon request is larger than a requested memory block:

allocating a portion of the determined segment large enough to satisfy the request;

providing a remaining portion as a new free memory segment; and

updating the first and second doubly linked lists accordingly.

33. (Previously Presented) The method of claim 27, wherein each segment is associated with a header, thereby to form the first doubly linked list, each header including information indicative of the size of the associated segment, information indicative of a state (free, allocated) of the associated segment, and a pointer indicative of a memory address of the previous segment.

34. (Previously Presented) The method of claim 30, wherein a header associated with each free segment of a given size further includes a pointer indicative of a memory address of a previous and/or subsequent free segment of the same size, depending on availability of a previous and/or subsequent free segment of the same size and in accordance with the order of free segments of the same size in the LIFO list.

35. (Previously Presented) The method of claim 17, further comprising:

allocating free segments of the data memory to different levels according to size; and

providing a bitmap comprising different stages, wherein the bits of one stage are indicative of availability of free segments in said levels, and the bits of another stage are indicative of state and/or size and/or location of individual segments.

36. (Previously Presented) The method of claim 35, wherein the bits of one stage are associated with pointers indicative of a memory address of free segments.

37. (Previously Presented) The method of claim 35, further comprising:

updating the bitmap to reflect the allocation or release of memory segments.

38. (Previously Presented) The method of claim 17, including freeing and allocating segments of the data memory, the method comprising, when freeing a memory segment:

determining a state of memory segments adjacent to the memory segment to be freed; and

merging the memory segment to be freed with free adjacent memory segments.

39. (Cancelled)

40. (Previously Presented) An operating system for a computer, adapted to perform the method of claim 1.

41. (Original) The operating system of claim 40, wherein the operating system is a realtime operating system.

42. (Previously Presented) The operating system of claim 40, adapted to perform the method described above at task level.

43. (Previously Presented) The operating system of claim 40, adapted to perform the method described above at interrupt level.

44. (Previously Presented) A computer program adapted to perform the method of claim 1 when operated on a computer.

45. (Previously Presented) A storage medium having stored thereon a set of instructions, which when executed by a computer, performs the method of claim 1.

46. (Previously Presented) A computer system comprising the operating system of claim 40.

47. (Cancelled)

48. (Previously Presented) The method of claim 17, further comprising:

defining a different granule size for each level, wherein the size of each memory segment is related to the granule size of the respective level.